

# Package ‘backbone’

July 22, 2025

**Type** Package

**Title** Extracts the Backbone from Graphs

**Version** 2.1.4

**Description** An implementation of methods for extracting an unweighted unipartite graph (i.e. a backbone) from an unweighted unipartite graph, a weighted unipartite graph, the projection of an unweighted bipartite graph, or the projection of a weighted bipartite graph (Neal, 2022 <[doi:10.1371/journal.pone.0269137](https://doi.org/10.1371/journal.pone.0269137)>).

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 2.10)

**Imports** igraph, Matrix, methods, stats, Rcpp, utils,

**Suggests** knitr, rmarkdown, tinytest

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**URL** <https://www.zacharyneal.com/backbone>,  
<https://github.com/zpneal/backbone>

**BugReports** <https://github.com/zpneal/backbone/issues>

**NeedsCompilation** yes

**Author** Zachary Neal [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-3076-4995>>),  
Rachel Domagalski [ctb],  
Bruce Sagan [ctb],  
Karl Godard [ctb]

**Maintainer** Zachary Neal <[zpneal@msu.edu](mailto:zpneal@msu.edu)>

**Repository** CRAN

**Date/Publication** 2024-05-09 18:50:03 UTC

## Contents

|                                        |           |
|----------------------------------------|-----------|
| backbone . . . . .                     | 2         |
| backbone.extract . . . . .             | 3         |
| backbone.suggest . . . . .             | 4         |
| bicm . . . . .                         | 5         |
| disparity . . . . .                    | 6         |
| fastball . . . . .                     | 8         |
| fdsm . . . . .                         | 9         |
| fixedcol . . . . .                     | 11        |
| fixedfill . . . . .                    | 13        |
| fixedrow . . . . .                     | 15        |
| global . . . . .                       | 17        |
| lans . . . . .                         | 18        |
| logit . . . . .                        | 20        |
| mlf . . . . .                          | 21        |
| osdsm . . . . .                        | 23        |
| pb . . . . .                           | 25        |
| sdsd . . . . .                         | 26        |
| sparsify . . . . .                     | 28        |
| sparsify.with.geometric . . . . .      | 30        |
| sparsify.with.gspar . . . . .          | 31        |
| sparsify.with.hypergeometric . . . . . | 32        |
| sparsify.with.jaccard . . . . .        | 33        |
| sparsify.with.localdegree . . . . .    | 34        |
| sparsify.with.lspar . . . . .          | 35        |
| sparsify.with.meetmin . . . . .        | 36        |
| sparsify.with.quadrilateral . . . . .  | 37        |
| sparsify.with.simmelian . . . . .      | 38        |
| sparsify.with.skeleton . . . . .       | 39        |
| <b>Index</b>                           | <b>41</b> |

---

backbone

*backbone: Extracts the Backbone from Graphs*


---

## Description

Provides methods for extracting from an unweighted and sparse subgraph (i.e., a backbone) that contains only the most "important" edges in a weighted bipartite projection, a non-projection weighted network, or an unweighted network.

Available backbone extraction functions include:

- For weighted bipartite projections of weighted bipartite networks: [osdsm\(\)](#).
- For weighted bipartite projections of binary bipartite networks: [fixedfill\(\)](#), [fixedrow\(\)](#), [fixedcol\(\)](#), [sdsd\(\)](#), and [fdsm\(\)](#).
- For non-projection weighted networks: [global\(\)](#), [disparity\(\)](#), [mlf\(\)](#), [lans\(\)](#).

- For unweighted networks: `sparsify()`, `sparsify.with.skeleton()`, `sparsify.with.gspar()`, `sparsify.with.lspar()`, `sparsify.with.simmelian()`, `sparsify.with.jaccard()`, `sparsify.with.meetmin()`, `sparsify.with.geometric()`, `sparsify.with.hypergeometric()`, `sparsify.with.localdegree()`, `sparsify.with.quadrilateral()`.
- For all networks: `backbone.suggest()` will examine the data and suggest an appropriate backbone function

The package also includes some utility functions:

- `fastball()` - Fast marginal-preserving randomization of binary matrices
- `bicm()` - Compute probabilities under the bipartite configuration model

For additional documentation and background on the package functions, see `vignette("backbone")`.

For updates, papers, presentations, and other backbone news, please see [www.rbackbone.net](http://www.rbackbone.net)

## References

Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:[10.1371/journal.pone.0269137](https://doi.org/10.1371/journal.pone.0269137)

---

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| backbone.extract | <i>Extracts a backbone network from a backbone object</i> |
|------------------|-----------------------------------------------------------|

---

## Description

`backbone.extract` returns a binary or signed adjacency matrix containing the backbone that retains only the significant edges.

## Usage

```
backbone.extract(
  bb.object,
  signed = FALSE,
  alpha = 0.05,
  mtc = "none",
  class = bb.object$class,
  narrative = FALSE
)
```

## Arguments

|                        |                                                                                                              |
|------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>bb.object</code> | backbone: backbone S3 class object.                                                                          |
| <code>signed</code>    | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                               |
| <code>alpha</code>     | real: significance level of hypothesis test(s)                                                               |
| <code>mtc</code>       | string: type of Multiple Test Correction to be applied; can be any method allowed by <code>p.adjust</code> . |

|           |                                                                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| class     | string: the class of the returned backbone graph, one of c("matrix", "sparseMatrix", "igraph", "edgelist"), converted via <a href="#">tomatrix</a> . |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                     |

## Details

The "backbone" S3 class object is composed of (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if signed = TRUE) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model

When signed = FALSE, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When signed = TRUE, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

## Value

backbone graph: Binary or signed backbone graph of class given in parameter class.

## Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10)),
  cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10)),
  cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10)))

backbone.object <- fixedrow(B, alpha = NULL)
bb <- backbone.extract(backbone.object, alpha = 0.05)
```

---

|                  |                                 |
|------------------|---------------------------------|
| backbone.suggest | <i>Suggest a backbone model</i> |
|------------------|---------------------------------|

---

## Description

backbone.suggest suggests and optionally runs an appropriate backbone model for a graph object.

## Usage

```
backbone.suggest(G, s = NULL)
```

Arguments

- G graph: A graph represented in an object of class matrix, sparse [Matrix](#), dataframe, or [igraph](#).
- s numeric: If provided, a backbone is extracted using this value as the significance level or sparsification parameter.

Value

If s == NULL: NULL, but a message is displayed with a suggested model. If 0 <= s <= 1: A binary backbone graph in the same class as G, obtained by extracting the backbone at the s significance level (if a statistical model is suggested) or using sparsification parameter s (if a sparsification model is suggested). The code used to perform the extraction, and suggested manuscript text are displayed.

References

Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:[10.1371/journal.pone.0269137](#)

Examples

```
M <- matrix(runif(100),10,10) #A random weighted, directed graph
backbone <- backbone.suggest(M)
backbone <- backbone.suggest(M, s = 0.05)
```

---

|      |                                      |
|------|--------------------------------------|
| bicm | <i>Bipartite Configuration Model</i> |
|------|--------------------------------------|

---

Description

bicm estimates cell probabilities under the bipartite configuration model

Usage

```
bicm(M, fitness = FALSE, tol = 1e-08, max_steps = 200, ...)
```

Arguments

- M matrix: a binary matrix
- fitness boolean: FALSE returns a matrix of probabilities, TRUE returns a list of row and column fitnesses only
- tol numeric, tolerance of algorithm
- max\_steps numeric, number of times to run [loglikelihood\\_prime\\_bicm](#) algorithm
- ... optional arguments

## Details

Given a binary matrix **M**, the Bipartite Configuration Model (BiCM; Saracco et. al. 2015) returns a valued matrix **B** in which  $B_{ij}$  is the *approximate* probability that  $M_{ij} = 1$  in the space of all binary matrices with the same row and column marginals as **M**. The BiCM yields the closest approximations of the true probabilities compared to other estimation methods (Neal et al., 2021), and is used by `sdsbm()` to extract the backbone of a bipartite projection using the stochastic degree sequence model.

Matrix **M** is "conforming" if no rows and no columns contain only zeros or only ones. If **M** is conforming, then `bicm()` is faster. Additionally, if `fitness = TRUE`, then `bicm()` returns a list of row and column fitnesses, which requires less memory. Given the  $i$ th row's fitness  $R_i$  and the  $j$ th column's fitness  $R_j$ , the entry  $B_{ij}$  in the probability matrix can be computed as  $R_i \times R_j / (1 + (R_i \times R_j))$ .

Matrix **M** is "non-conforming" if any rows or any columns contain only zeros or only ones. If **M** is non-conforming, then `bicm()` is slower and will only return a probability matrix.

## Value

a matrix of probabilities or a list of fitnesses

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

bicm: Saracco, F., Di Clemente, R., Gabrielli, A., & Squartini, T. (2015). Randomizing bipartite networks: The case of the World Trade Web. *Scientific Reports*, 5, 10595. doi:10.1038/srep10595

## Examples

```
M <- matrix(c(0,0,1,0,1,0,1,0,1),3,3) #A binary matrix
bicm(M)
```

---

disparity

*Extract backbone using the Disparity Filter*

---

## Description

`disparity` extracts the backbone of a weighted network using the Disparity Filter.

## Usage

```
disparity(
  W,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
```

```

    class = "original",
    narrative = FALSE
  )

```

## Arguments

|                              |                                                                                                                                                                                                                               |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>W</code>               | A positively-weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a three-column dataframe; (3) an <a href="#">igraph</a> object. |
| <code>alpha</code>           | real: significance level of hypothesis test(s)                                                                                                                                                                                |
| <code>missing.as.zero</code> | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                                |
| <code>signed</code>          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                                |
| <code>mtc</code>             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                               |
| <code>class</code>           | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as <code>W</code> .           |
| <code>narrative</code>       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                              |

## Details

The `disparity` function applies the disparity filter (Serrano et al., 2009), which compares an edge's weight to its expected weight if a node's total degree was uniformly distributed across all its edges. The graph may be directed or undirected, however the edge weights must be positive.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

If `W` is an unweighted bipartite graph, then the disparity filter is applied to its weighted bipartite projection.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using [backbone.extract\(\)](#).

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

disparity filter: Serrano, M. A., Boguna, M., & Vespignani, A. (2009). Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106, 6483-6488. doi:10.1073/pnas.0808904106

## Examples

```
#A network with heterogeneous (i.e. multiscale) weights
net <- matrix(c(0,10,10,10,10,10,75,0,0,0,0,0,
               10,0,1,1,1,0,0,0,0,0,0,
               10,1,0,1,1,0,0,0,0,0,0,
               10,1,1,0,1,0,0,0,0,0,0,
               10,1,1,1,0,0,0,0,0,0,0,
               75,0,0,0,0,0,100,100,100,100,
               0,0,0,0,0,100,0,10,10,10,
               0,0,0,0,0,100,10,0,10,10,
               0,0,0,0,0,100,10,10,0,10,
               0,0,0,0,0,100,10,10,10,0),10)

net <- igraph::graph_from_adjacency_matrix(net, mode = "undirected", weighted = TRUE)
plot(net, edge.width = sqrt(igraph::E(net)$weight)) #A stronger clique & a weaker clique

strong <- igraph::delete_edges(net, which(igraph::E(net)$weight < mean(igraph::E(net)$weight)))
plot(strong) #A backbone of stronger-than-average edges ignores the weaker clique

bb <- disparity(net, alpha = 0.05, narrative = TRUE) #A disparity backbone...
plot(bb) #...preserves edges at multiple scales
```

---

fastball

Randomize a binary matrix using the fastball algorithm

---

## Description

fastball randomizes a binary matrix, preserving the row and column sums

## Usage

```
fastball(M, trades = 5 * nrow(M))
```

## Arguments

|        |                                                                           |
|--------|---------------------------------------------------------------------------|
| M      | matrix: a binary matrix (see details)                                     |
| trades | integer: number of trades; the default is 5R trades (approx. mixing time) |

## Details

Given a matrix M, fastball randomly samples a new matrix from the space of all matrices with the same row and column sums as M.

## Value

matrix: A random binary matrix with same row sums and column sums as M.

## References

fastball: Godard, Karl and Neal, Zachary P. 2022. fastball: A fast algorithm to sample bipartite graphs with fixed degree sequences. *Journal of Complex Networks* doi:10.1093/comnet/cnac049

## Examples

```
M <- matrix(rbinom(200,1,0.5),10,20) #A random 10x20 binary matrix
Mrand <- fastball(M) #Random matrix with same row and column sums
```

---

 fsm

---

*Extract backbone using the Fixed Degree Sequence Model*


---

## Description

fdsm extracts the backbone of a bipartite projection using the Fixed Degree Sequence Model.

## Usage

```
fdsm(
  B,
  alpha = 0.05,
  trials = NULL,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE,
  progress = TRUE,
  ...
)
```

## Arguments

|       |                                                                                                                                                                                                                    |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B     | An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha | real: significance level of hypothesis test(s)                                                                                                                                                                     |

|                              |                                                                                                                                                                                                                     |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>trials</code>          | numeric: the number of bipartite graphs generated to approximate the edge weight distribution. If NULL, the number of trials is selected based on <code>alpha</code> (see details)                                  |
| <code>missing.as.zero</code> | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                      |
| <code>signed</code>          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                      |
| <code>mtc</code>             | string: type of Multiple Test Correction to be applied; can be any method allowed by <code>p.adjust</code> .                                                                                                        |
| <code>class</code>           | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as <code>B</code> . |
| <code>narrative</code>       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |
| <code>progress</code>        | boolean: TRUE if the progress of Monte Carlo trials should be displayed.                                                                                                                                            |
| <code>...</code>             | optional arguments                                                                                                                                                                                                  |

## Details

The `fdsm` function compares an edge's observed weight in the projection  $B \times t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite network where both the row vertex degrees and column vertex degrees are *exactly* fixed at their values in `B`. It uses the `fastball()` algorithm to generate random bipartite matrices with given row and column vertex degrees.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

The p-values used to evaluate the statistical significance of each edge are computed using Monte Carlo methods. The number of trials performed affects the precision of these p-values. This precision impacts the confidence that a given edge's p-value is less than the desired alpha level, and therefore represents a statistically significant edge that should be retained in the backbone. When `trials = NULL`, `trials.needed()` is used to estimate the required number of trials to evaluate the statistical significance of an edge's p-values.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using `backbone.extract()`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

fdsm: Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*. doi:10.1038/s41598021032383

fastball: Godard, Karl and Neal, Zachary P. 2022. fastball: A fast algorithm to sample bipartite graphs with fixed degree sequences. *Journal of Complex Networks* doi:10.1093/comnet/cnac049

## Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fdsm(B, alpha = 0.05, trials = 1000, narrative = TRUE, class = "igraph") #An FDSM backbone...
plot(bb) #...is sparse with clear communities
```

---

fixedcol

---

*Extract backbone using the Fixed Column Model*


---

## Description

fixedcol extracts the backbone of a bipartite projection using the Fixed Column Model.

## Usage

```
fixedcol(
  B,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

## Arguments

|                 |                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B               | An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                     |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                     |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                     |
| mtc             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                    |
| class           | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as B.                            |
| narrative       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                   |

## Details

This `fixedcol` function compares an edge's observed weight in the projection  $B * t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite graph where the *column* vertex degrees are fixed but the row vertex degrees are allowed to vary.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using [backbone.extract\(\)](#).

## References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:[10.1371/journal.pone.0269137](https://doi.org/10.1371/journal.pone.0269137)
- fixedcol: Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11, 23929. doi:[10.1038/s41598021032383](https://doi.org/10.1038/s41598021032383)

## Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fixedcol(B, alpha = 0.05, narrative = TRUE, class = "igraph") #A fixedcol backbone...
plot(bb) #...is sparse with clear communities
```

---

fixedfill

---

*Extract backbone using the Fixed Fill Model*


---

## Description

fixedfill extracts the backbone of a bipartite projection using the Fixed Fill Model.

## Usage

```
fixedfill(
  B,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

## Arguments

|                 |                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B               | An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                     |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                     |

|           |                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| signed    | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                          |
| mtc       | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                         |
| class     | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as B. |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                        |

## Details

The `fixedfill` function compares an edge's observed weight in the projection  $B * t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite graph where the number of edges present (i.e., the number of cells *filled* with a 1) is equal to the number of edges in B. When B is large, this function may be impractically slow and may return a backbone object that contains NaN values.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using [backbone.extract\(\)](#).

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

fixedfill: Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11, 23929. doi:10.1038/s41598021032383

## Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10)),
          cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10)),
```

```

cbind(matrix(rbinom(250,1,.2),10),
       matrix(rbinom(250,1,.2),10),
       matrix(rbinom(250,1,.8),10))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fixedfill(B, alpha = 0.05, narrative = TRUE, class = "igraph") #A fixedfill backbone...
plot(bb) #...is sparse with clear communities

```

fixedrow

*Extract backbone using the Fixed Row Model*

## Description

fixedrow extracts the backbone of a bipartite projection using the Fixed Row Model.

## Usage

```

fixedrow(
  B,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)

```

## Arguments

|                 |                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B               | An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                     |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                     |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                     |
| mtc             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                    |
| class           | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as B.                            |
| narrative       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                   |

## Details

The `fixedrow` function compares an edge's observed weight in the projection  $B * t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite graph where the *row* vertex degrees are fixed but the column vertex degrees are allowed to vary.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using `backbone.extract()`.

## References

- package: Neal, Z. P. (2022). `backbone`: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137
- Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11, 23929. doi:10.1038/s41598021032383

## Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10),
                    matrix(rbinom(250,1,.2),10)),
           cbind(matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.2),10),
                    matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- fixedrow(B, alpha = 0.05, narrative = TRUE, class = "igraph") #A fixedrow backbone...
plot(bb) #...is sparse with clear communities
```

---

|        |                                          |
|--------|------------------------------------------|
| global | <i>Compute global threshold backbone</i> |
|--------|------------------------------------------|

---

## Description

global extracts the backbone of a weighted network using a global threshold

## Usage

```
global(
  G,
  upper = 0,
  lower = NULL,
  keepzeros = TRUE,
  class = "original",
  narrative = FALSE
)
```

## Arguments

|           |                                                                                                                                                                                                                                  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G         | A weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> , or dataframe; (2) an edgelist in the form of a three-column dataframe; (3) an <a href="#">igraph</a> object. |
| upper     | real, FUN, or NULL: upper threshold value or function that evaluates to an upper threshold value.                                                                                                                                |
| lower     | real, FUN, or NULL: lower threshold value or function that evaluates to a lower threshold value.                                                                                                                                 |
| keepzeros | boolean: TRUE if zero-weight edges in W should be excluded from (i.e. also be zero in) the backbone                                                                                                                              |
| class     | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as W.                                          |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                                 |

## Details

The global function retains a edge with weight  $W$  if  $W > \text{upper}$ . If a lower threshold is also specified, it returns a signed backbone in which an edge's weight is set to 1 if  $W > \text{upper}$ , is set to -1 if  $W < \text{lower}$ , and is set to 0 otherwise. The default is an unsigned backbone containing all edges with non-zero weights.

If G is an unweighted bipartite graph, the global threshold is applied to its weighted bipartite projection.

## Value

Binary or signed backbone graph of class given in parameter class.

## References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137
- model: Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance, and other co-behaviors. *Social Networks*, 39, 84-97. doi:10.1016/j.socnet.2014.06.001

## Examples

```
G <- matrix(sample(0:5, 100, replace = TRUE), 10) #Random weighted graph
diag(G) <- 0
G
global(G, narrative = TRUE) #Keep all non-zero edges
global(G, upper = 4, lower = 2, narrative = TRUE) #Signed with specified thresholds
global(G, upper = function(x)mean(x), #Above-average --> positive edges
        lower = function(x)mean(x), narrative = TRUE) #Below-average --> negative edges
```

---

lans

---

*Extract backbone using Locally Adaptive Network Sparsification*


---

## Description

lans extracts the backbone of a weighted network using Locally Adaptive Network Sparsification

## Usage

```
lans(
  W,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

## Arguments

- |                 |                                                                                                                                                                                                                               |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| W               | A positively-weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a three-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                                |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                                |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                                |

|           |                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mtc       | string: type of Multiple Test Correction to be applied; can be any method allowed by <code>p.adjust</code> .                                                                                                        |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as <code>W</code> . |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

## Details

The `lans` function applies Locally Adaptive Network Sparsification (LANS; Foti et al., 2011), which compares an edge's fractional weight to the cumulative distribution function for the fractional edge weights of all edges connected to a given node. The graph may be directed or undirected, however the edge weights must be positive.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

If `W` is an unweighted bipartite graph, then LANS is applied to its weighted bipartite projection.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using `backbone.extract()`.

## References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137
- lans: Foti, N. J., Hughes, J. M., and Rockmore, D. N. (2011). Nonparametric Sparsification of Complex Multiscale Networks. *PLOS One*, 6, e16431. doi:10.1371/journal.pone.0016431

## Examples

```
#Simple star from Foti et al. (2011), Figure 2
net <- matrix(c(0,2,2,2,2,
               2,0,1,1,0,
               2,1,0,0,1,
               2,1,0,0,1,
               2,0,1,1,0),5,5)
net <- igraph::graph_from_adjacency_matrix(net, mode = "undirected", weighted = TRUE)
plot(net, edge.width = igraph::E(net)$weight^2)

bb <- lans(net, alpha = 0.05, narrative = TRUE) #The LANS backbone
plot(bb)
```

logit

*Logit-based probabilities for SDSM***Description**

logit estimates cell probabilities under the logit model

**Usage**

```
logit(M)
```

**Arguments**

**M**                      matrix

**Details**

Given a matrix **M**, the logit model returns a valued matrix **B** in which  $B_{ij}$  is the *approximate* probability that  $M_{ij} = 1$  in the space of all binary matrices with the same row and column marginals as **M**.

The Bipartite Configuration Model (BiCM), which is available using [bicm](#) is faster and yields slightly more accurate probabilities (Neal et al., 2021). Therefore, it is the default used in [sds](#). However, the BiCM it requires the assumption that any cell in **M** can take a value of 0 or 1.

In contrast, the logit model allows constraints on specific cells. If **M** represents a bipartite graph, these constraints are equivalent to structural 0s (an edge that can never be present) and structural 1s (an edge that must always be present). To impose such constraints, **M** should be binary, except that structural 0s are represented with  $M_{ij} = 10$ , and structural 1s are represented with  $M_{ij} = 11$ .

**Value**

a matrix of probabilities

**References**

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

logit model: Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance and other co-behaviors. *Social Networks*, 39, 84-97. doi:10.1016/j.socnet.2014.06.001

logit model with constraints: Neal, Z. P. and Neal, J. W. (2024). Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC) for Backbone Extraction. *Proceedings of the 12th International Conference on Complex Networks and their Applications*. Springer.

## Examples

```
M <- matrix(c(0,0,1,0,1,0,1,0,1),3,3) #A binary matrix
logit(M)
M <- matrix(c(0,10,1,0,1,0,1,0,11),3,3) #A binary matrix with structural values
logit(M)
```

mlf

*Extract backbone using the Marginal Likelihood Filter*

## Description

mlf extracts the backbone of a weighted network using the Marginal Likelihood Filter

## Usage

```
mlf(
  W,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE
)
```

## Arguments

|                 |                                                                                                                                                                                                                             |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| W               | An integer-weighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a three-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                              |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                              |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                              |
| mtc             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                             |
| class           | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as W.                                     |
| narrative       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                            |

## Details

The `mlf` function applies the marginal likelihood filter (MLF; Dianati, 2016), which compares an edge's weight to its expected weight in a graph that preserves the total weight and preserves the degree sequence *on average*. The graph may be directed or undirected, however the edge weights must be positive integers.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

If `W` is an unweighted bipartite graph, then the MLF is applied to its weighted bipartite projection.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using `backbone.extract()`.

## References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:[10.1371/journal.pone.0269137](https://doi.org/10.1371/journal.pone.0269137)
- mlf: Dianati, N. (2016). Unwinding the hairball graph: Pruning algorithms for weighted complex networks. *Physical Review E*, 93, 012304. doi:[10.1103/PhysRevE.93.012304](https://doi.org/10.1103/PhysRevE.93.012304)

## Examples

```
#A network with heterogeneous weights
net <- matrix(c(0,10,10,10,10,75,0,0,0,0,0,
               10,0,1,1,1,0,0,0,0,0,0,
               10,1,0,1,1,0,0,0,0,0,0,
               10,1,1,0,1,0,0,0,0,0,0,
               10,1,1,1,0,0,0,0,0,0,0,
               75,0,0,0,0,0,100,100,100,100,
               0,0,0,0,0,100,0,10,10,10,
               0,0,0,0,0,100,10,0,10,10,
               0,0,0,0,0,100,10,10,0,10,
               0,0,0,0,0,100,10,10,10,0),10)

net <- igraph::graph_from_adjacency_matrix(net, mode = "undirected", weighted = TRUE)
plot(net, edge.width = sqrt(igraph::E(net)$weight)) #A stronger clique & a weaker clique

strong <- igraph::delete_edges(net, which(igraph::E(net)$weight < mean(igraph::E(net)$weight)))
plot(strong) #A backbone of stronger-than-average edges ignores the weaker clique

bb <- mlf(net, alpha = 0.05, narrative = TRUE) #An MLF backbone...
```

```
plot(bb) #...preserves edges at multiple scales
```

---

|       |                                                                            |
|-------|----------------------------------------------------------------------------|
| osdsm | <i>Extract backbone using the Ordinal Stochastic Degree Sequence Model</i> |
|-------|----------------------------------------------------------------------------|

---

## Description

osdsm extracts the backbone of a bipartite projection using the Ordinal Stochastic Degree Sequence Model.

## Usage

```
osdsm(
  B,
  alpha = 0.05,
  trials = NULL,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE,
  progress = TRUE,
  ...
)
```

## Arguments

|                 |                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B               | An ordinally weighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a three-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                               |
| trials          | integer: the number of bipartite graphs generated to approximate the edge weight distribution. If NULL, the number of trials is selected based on alpha (see details)                                                        |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                               |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                               |
| mtc             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                              |
| class           | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as B.                                      |

|           |                                                                          |
|-----------|--------------------------------------------------------------------------|
| narrative | boolean: TRUE if suggested text & citations should be displayed.         |
| progress  | boolean: TRUE if the progress of Monte Carlo trials should be displayed. |
| ...       | optional arguments                                                       |

## Details

The `osdsm` function compares an edge's observed weight in the projection  $B \times t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite network where both the rows and the columns contain approximately the same number of each value. The edges in  $B$  must be integers, and are assumed to represent an ordinal-level measure such as a Likert scale that starts at 0.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

The p-values used to evaluate the statistical significance of each edge are computed using Monte Carlo methods. The number of trials performed affects the precision of these p-values. This precision impacts the confidence that a given edge's p-value is less than the desired alpha level, and therefore represents a statistically significant edge that should be retained in the backbone. When `trials = NULL`, `trials.needed()` is used to estimate the required number of trials to evaluate the statistical significance of an edges' p-values.

## Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using `backbone.extract()`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

osdsm: Neal, Z. P. (2017). Well connected compared to what? Rethinking frames of reference in world city network research. *Environment and Planning A*, 49, 2859-2877. doi:10.1177/0308518X16631339

## Examples

```
#A weighted binary bipartite network of 20 agents & 50 artifacts; agents form two communities
B <- rbind(cbind(matrix(sample(0:3, 250, replace = TRUE, prob = ((1:4)^2)),10),
  matrix(sample(0:3, 250, replace = TRUE, prob = ((4:1)^2)),10)),
  cbind(matrix(sample(0:3, 250, replace = TRUE, prob = ((4:1)^2)),10),
  matrix(sample(0:3, 250, replace = TRUE, prob = ((1:4)^2)),10)))
```

```

P <- B%*%t(B) #An ordinary weighted projection...
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- osdsm(B, alpha = 0.05, narrative = TRUE, #An oSDSM backbone...
            class = "igraph", trials = 100)
plot(bb) #...is sparse with clear communities

```

pb

*Poisson binomial distribution function***Description**

pb computes the poisson binomial distribution function using the refined normal approximation.

**Usage**

```
pb(k, p, lowertail = TRUE)
```

**Arguments**

|           |                                                                                                            |
|-----------|------------------------------------------------------------------------------------------------------------|
| k         | numeric: value where the pdf should be evaluated                                                           |
| p         | vector: vector of success probabilities                                                                    |
| lowertail | boolean: If TRUE return both upper & lower tail probabilities, if FALSE return only upper tail probability |

**Details**

The Refined Normal Approximation (RNA) offers a close approximation when  $\text{length}(p)$  is large (Hong, 2013).

**Value**

vector, length 2: The first value (if lower = TRUE) is the lower tail probability, the probability of observing k or fewer successes when each trial has probability p of success. The second value is the upper tail probability, the probability of observing k or more successes when each trial has probability p of success.

**References**

Hong, Y. (2013). On computing the distribution function for the Poisson binomial distribution. *Computational Statistics and Data Analysis*, 59, 41-51. doi:[10.1016/j.csda.2012.10.006](https://doi.org/10.1016/j.csda.2012.10.006)

**Examples**

```
pb(50, runif(100))
```

sdsd

*Extract backbone using the Stochastic Degree Sequence Model*

## Description

sdsd extracts the backbone of a bipartite projection using the Stochastic Degree Sequence Model.

## Usage

```
sdsd(
  B,
  alpha = 0.05,
  missing.as.zero = FALSE,
  signed = FALSE,
  mtc = "none",
  class = "original",
  narrative = FALSE,
  ...
)
```

## Arguments

|                 |                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| B               | An unweighted bipartite graph, as: (1) an incidence matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| alpha           | real: significance level of hypothesis test(s)                                                                                                                                                                     |
| missing.as.zero | boolean: should missing edges be treated as edges with zero weight and tested for significance                                                                                                                     |
| signed          | boolean: TRUE for a signed backbone, FALSE for a binary backbone (see details)                                                                                                                                     |
| mtc             | string: type of Multiple Test Correction to be applied; can be any method allowed by <a href="#">p.adjust</a> .                                                                                                    |
| class           | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as B.                            |
| narrative       | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                   |
| ...             | optional arguments                                                                                                                                                                                                 |

## Details

The sdsd function compares an edge's observed weight in the projection  $B \times t(B)$  to the distribution of weights expected in a projection obtained from a random bipartite network where both the row vertex degrees and column vertex degrees are *approximately* fixed at their values in B.

When `signed = FALSE`, a one-tailed test (is the weight stronger?) is performed for each edge. The resulting backbone contains edges whose weights are significantly *stronger* than expected in the null model. When `signed = TRUE`, a two-tailed test (is the weight stronger or weaker?) is performed for each edge. The resulting backbone contains positive edges for those whose weights are significantly *stronger*, and negative edges for those whose weights are significantly *weaker*, than expected in the null model.

The bipartite network `B` may contain some edges that are *required* in the null model (i.e., structural 1s); these edges should have a weight of 11 (i.e., `B_ik = 11`). This network may also contain some edges that are *prohibited* in the null model (i.e., structural 0s); these edges should have a weight of 10 (i.e., `B_ik = 10`). When `B` contains required or prohibited edges, cellwise probabilities are computed using [logit](#) following Neal et al. (2024). Otherwise, cellwise probabilities are computed using the faster and more accurate Bipartite Configuration Model with [bicm](#) (Neal et al. 2021).

### Value

If `alpha != NULL`: Binary or signed backbone graph of class `class`.

If `alpha == NULL`: An S3 backbone object containing (1) the weighted graph as a matrix, (2) upper-tail p-values as a matrix, (3, if `signed = TRUE`) lower-tail p-values as a matrix, (4, if present) node attributes as a dataframe, and (5) several properties of the original graph and backbone model, from which a backbone can subsequently be extracted using [backbone.extract\(\)](#).

### References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137
- sdsdm: Neal, Z. P. (2014). The backbone of bipartite projections: Inferring relationships from co-authorship, co-sponsorship, co-attendance, and other co-behaviors. *Social Networks*, 39, 84-97. doi:10.1016/j.socnet.2014.06.001
- bicm: Neal, Z. P., Domagalski, R., and Sagan, B. (2021). Comparing Alternatives to the Fixed Degree Sequence Model for Extracting the Backbone of Bipartite Projections. *Scientific Reports*, 11, 23929. doi:10.1038/s41598021032383
- logit: Neal, Z. P. and Neal, J. W. (2024). Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC) for Backbone Extraction. *Proceedings of the 12th International Conference on Complex Networks and their Applications*. Springer.

### Examples

```
#A binary bipartite network of 30 agents & 75 artifacts; agents form three communities
B <- rbind(cbind(matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10)),
  cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10),
  matrix(rbinom(250,1,.2),10)),
  cbind(matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.2),10),
  matrix(rbinom(250,1,.8),10)))

P <- B%*%t(B) #An ordinary weighted projection...
```

```
plot(igraph::graph_from_adjacency_matrix(P, mode = "undirected",
                                         weighted = TRUE, diag = FALSE)) #...is a dense hairball

bb <- sds(B, alpha = 0.05, narrative = TRUE, class = "igraph") #An SDSM backbone...
plot(bb) #...is sparse with clear communities
```

---

sparsify

---

*Extract the backbone from a network using a sparsification model*


---

## Description

A generic function to extract the backbone of an undirected, unipartite network using a sparsification model described by a combination of an edge scoring metric, a edge score normalization, and an edge score filter.

## Usage

```
sparsify(
  U,
  s,
  escore,
  normalize,
  filter,
  symmetrize = TRUE,
  umst = FALSE,
  class = "original",
  narrative = FALSE
)
```

## Arguments

|            |                                                                                                                                                                                                                     |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U          | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| s          | numeric: Sparsification parameter                                                                                                                                                                                   |
| escore     | string: Method for scoring edges' importance                                                                                                                                                                        |
| normalize  | string: Method for normalizing edge scores                                                                                                                                                                          |
| filter     | string: Type of filter to apply                                                                                                                                                                                     |
| symmetrize | boolean: TRUE if the result should be symmetrized                                                                                                                                                                   |
| umst       | boolean: TRUE if the backbone should include the union of minimum spanning trees, to ensure connectivity                                                                                                            |
| class      | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as U.                             |
| narrative  | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

## Details

The `escore` parameter determines how an unweighted edge's importance is calculated. Unless noted below, scores are symmetric and larger values represent more important edges.

- `random`: a random number drawn from a uniform distribution
- `betweenness`: edge betweenness
- `triangles`: number of triangles that include the edge
- `jaccard`: jaccard similarity coefficient of the neighborhoods of an edge's endpoints, or alternatively, triangles normalized by the size of the union of the endpoints neighborhoods
- `dice`: dice similarity coefficient of the neighborhoods of an edge's endpoints
- `quadrangles`: number of quadrangles that include the edge
- `quadrilateral_embeddedness`: geometric mean normalization of quadrangles
- `degree`: degree of neighbor to which an edge is adjacent (asymmetric)
- `meetmin`: triangles normalized by the smaller of the endpoints' neighborhoods' sizes
- `geometric`: triangles normalized by the product of the endpoints' neighborhoods' sizes
- `hypergeometric`: probability of the edge being included at least as many triangles if edges were random, given the size of the endpoints' neighborhoods (smaller is more important)

The `normalize` parameter determines whether edge scores are normalized.

- `none`: no normalization is performed
- `rank`: scores are normalized by neighborhood rank, such that the strongest edge in a node's neighborhood is ranked 1 (asymmetric)
- `embeddedness`: scores are normalized using the maximum Jaccard coefficient of the top `k`-ranked neighbors of each endpoint, for all `k`

The `filter` parameter determines how edges are filtered based on their (normalized) edge scores.

- `threshold`: Edges with scores  $\geq s$  are retained in the backbone
- `proportion`: Specifies the approximate proportion of edges to retain in the backbone
- `degree`: Retains each node's  $d^s$  most important edges, where  $d$  is the node's degree (requires that `normalize = "rank"`)
- `disparity`: Applies the disparity filter using `disparity()`

Using `escore == "degree"` or `normalize == "rank"` can yield an asymmetric network. When `symmetrize == TRUE` (default), after applying a filter, the network is symmetrized by such that  $i \rightarrow j$  if  $i \rightarrow j$  or  $i \leftarrow j$ .

Specific combinations of `escore`, `normalize`, `filter`, and `umst` correspond to specific sparsification models in the literature, and are available via the following wrapper functions: `sparsify.with.skeleton()`, `sparsify.with.gspar()`, `sparsify.with.lspar()`, `sparsify.with.simmelian()`, `sparsify.with.jaccard()`, `sparsify.with.meetmin()`, `sparsify.with.geometric()`, `sparsify.with.hypergeometric()`, `sparsify.with.localdegree()`, `sparsify.with.quadrilateral()`. See the documentation for these wrapper functions for more details and the associated citation.

**Value**

An unweighted, undirected, unipartite graph of class `class`.

**References**

Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

**Examples**

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify(U, s = 0.6, escore = "jaccard", normalize = "rank",
  filter = "degree", narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.geometric

*Extract Goldberg and Roth's (2003) Geometric backbone*

---

**Description**

`sparsify.with.geometric` is a wrapper for `sparsify()` that extracts the geometric backbone described by Goldberg and Roth (2003). It is equivalent to `sparsify(escore = "geometric", normalize = "none", filter = "threshold", umst = FALSE)`.

**Usage**

```
sparsify.with.geometric(U, s, class = "original", narrative = FALSE)
```

**Arguments**

|                        |                                                                                                                                                                                                                     |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>U</code>         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object.       |
| <code>s</code>         | numeric: Sparsification threshold, $0 < s < 1$ ; larger values yield sparser graphs                                                                                                                                 |
| <code>class</code>     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as <code>U</code> . |
| <code>narrative</code> | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

**Value**

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Goldberg, D. S., & Roth, F. P. (2003). Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100, 4372-4376. doi:10.1073/pnas.0735871100

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.geometric(U, s = 0.25, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| sparsify.with.gspar | <i>Extract Satuluri et al's (2011) G-spar backbone</i> |
|---------------------|--------------------------------------------------------|

---

## Description

sparsify.with.gspar is a wrapper for `sparsify()` that extracts the G-spar backbone described by Satuluri et al. (2011). It is equivalent to `sparsify(escape = "jaccard", normalize = "none", filter = "proportion", umst = FALSE)`.

## Usage

```
sparsify.with.gspar(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Proportion of edges to retain, $0 < s < 1$ ; smaller values yield sparser graphs                                                                                                                     |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Satuluri, V., Parthasarathy, S., & Ruan, Y. (2011, June). Local graph sparsification for scalable clustering. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 721-732). doi:10.1145/1989323.1989399

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.gspar(U, s = 0.4, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.hypergeometric

*Extract Goldberg and Roth's (2003) Hypergeometric backbone*

---

## Description

sparsify.with.hypergeometric is a wrapper for `sparsify()` that extracts the hypergeometric backbone described by Goldberg and Roth (2003). It is equivalent to `sparsify(escape = "hypergeometric", normalize = "none", filter = "threshold", umst = FALSE)`.

## Usage

```
sparsify.with.hypergeometric(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Sparsification threshold, $0 < s < 1$ ; smaller values yield sparser graphs                                                                                                                          |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Goldberg, D. S., & Roth, F. P. (2003). Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100, 4372-4376. doi:10.1073/pnas.0735871100

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.hypergeometric(U, s = 0.3, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.jaccard *Extract Goldberg and Roth's (2003) Jaccard backbone*

---

## Description

sparsify.with.jaccard is a wrapper for [sparsify\(\)](#) that extracts the jaccard backbone described by Goldberg and Roth (2003). It is equivalent to `sparsify(escape = "jaccard", normalize = "none", filter = "threshold", umst = FALSE)`.

## Usage

```
sparsify.with.jaccard(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| s         | numeric: Sparsification threshold, $0 < s < 1$ ; larger values yield sparser graphs                                                                                                                                 |
| class     | string: the class of the returned backbone graph, one of c("original", "matrix", "Matrix", "igraph", "edgelist"). If "original", the backbone graph returned is of the same class as U.                             |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

## Value

An unweighted, undirected, unipartite graph of class class.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Goldberg, D. S., & Roth, F. P. (2003). Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100, 4372-4376. doi:10.1073/pnas.0735871100

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.jaccard(U, s = 0.3, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.localdegree

*Extract Hamann et al.'s (2016) Local Degree backbone*

---

## Description

sparsify.with.localdegree is a wrapper for `sparsify()` that extracts the local degree backbone described by Hamann et al. (2016). It is equivalent to `sparsify(escore = "degree", normalize = "rank", filter = "degree", umst = FALSE)`.

## Usage

```
sparsify.with.localdegree(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Sparsification exponent, $0 < s < 1$ ; smaller values yield sparser graphs                                                                                                                           |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Hamann, M., Lindner, G., Meyerhenke, H., Staudt, C. L., & Wagner, D. (2016). Structure-preserving sparsification methods for social networks. *Social Network Analysis and Mining*, 6, 22. doi:10.1007/s1327801603322

## Examples

```
U <- igraph::as.undirected(igraph::sample_pa(60, m = 3), mode = "collapse")
plot(U) #A hairball
sparse <- sparsify.with.localdegree(U, s = 0.3, narrative = TRUE)
plot(sparse) #Clearly visible hubs
```

---

sparsify.with.lspar      *Extract Satuluri et al's (2011) L-spar backbone*

---

## Description

sparsify.with.lspar is a wrapper for [sparsify\(\)](#) that extracts the L-spar backbone described by Satuluri et al. (2011). It is equivalent to `sparsify(escore = "jaccard", normalize = "rank", filter = "degree", umst = FALSE)`.

## Usage

```
sparsify.with.lspar(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| s         | numeric: Sparsification exponent, $0 < s < 1$ ; smaller values yield sparser graphs                                                                                                                                 |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.               |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Satuluri, V., Parthasarathy, S., & Ruan, Y. (2011, June). Local graph sparsification for scalable clustering. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 721-732). doi:10.1145/1989323.1989399

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.lspar(U, s = 0.6, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.meetmin *Extract Goldberg and Roth's (2003) MeetMin backbone*

---

## Description

sparsify.with.meetmin is a wrapper for `sparsify()` that extracts the meetmin backbone described by Goldberg and Roth (2003). It is equivalent to `sparsify(escore = "meetmin", normalize = "none", filter = "threshold", umst = FALSE)`.

## Usage

```
sparsify.with.meetmin(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Sparsification threshold, $0 < s < 1$ ; larger values yield sparser graphs                                                                                                                           |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Goldberg, D. S., & Roth, F. P. (2003). Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100, 4372-4376. doi:10.1073/pnas.0735871100

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.meetmin(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.quadrilateral

*Extract Nocaj et al.'s (2015) Quadrilateral Simmelian backbone*

---

## Description

sparsify.with.quadrilateral is a wrapper for `sparsify()` that extracts the quadrilateral Simmelian backbone described by Nocaj et al. (2015). It is equivalent to `sparsify(escore = "quadrilateral embeddedness", normalize = "embeddedness", filter = "threshold", umst = TRUE)`.

## Usage

```
sparsify.with.quadrilateral(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Sparsification exponent, $0 < s < 1$ ; larger values yield sparser graphs                                                                                                                            |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Nocaj, A., Ortmann, M., & Brandes, U. (2015). Untangling the hairballs of multi-centered, small-world online social media networks. *Journal of Graph Algorithms and Applications*, 19, 595-618. doi:10.7155/jgaa.00370

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.quadrilateral(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities in a connected graph
```

---

sparsify.with.simmelian

*Extract Nick et al's (2013) Simmelian backbone*

---

## Description

sparsify.with.simmelian is a wrapper for [sparsify\(\)](#) that extracts the simmelian backbone described by Nick et al. (2013). It is equivalent to `sparsify(escore = "triangles", normalize = "embeddedness", filter = "threshold", umst = FALSE)`.

## Usage

```
sparsify.with.simmelian(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <a href="#">Matrix</a> ; (2) an edgelist in the form of a two-column dataframe; (3) an <a href="#">igraph</a> object. |
| s         | numeric: Sparsification threshold, $0 < s < 1$ ; larger values yield sparser graphs                                                                                                                                 |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.               |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                                    |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:10.1371/journal.pone.0269137

model: Nick, B., Lee, C., Cunningham, P., & Brandes, U. (2013, August). Simmelian backbones: Amplifying hidden homophily in facebook networks. In Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining (pp. 525-532). doi:10.1145/2492517.2492569

## Examples

```
U <- igraph::sample_sbm(60, matrix(c(.75,.25,.25,.25,.75,.25,.25,.25,.75),3,3), c(20,20,20))
plot(U) #A hairball
sparse <- sparsify.with.simmelian(U, s = 0.5, narrative = TRUE)
plot(sparse) #Clearly visible communities
```

---

sparsify.with.skeleton

*Extract Karger's (1999) skeleton backbone*

---

## Description

sparsify.with.skeleton is a wrapper for `sparsify()` that extracts the skeleton backbone described by Karger (1999), which preserves a specified proportion of random edges. It is equivalent to `sparsify(escore = "random", normalize = "none", filter = "proportion", umst = FALSE)`.

## Usage

```
sparsify.with.skeleton(U, s, class = "original", narrative = FALSE)
```

## Arguments

|           |                                                                                                                                                                                                               |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| U         | An unweighted unipartite graph, as: (1) an adjacency matrix in the form of a matrix or sparse <code>Matrix</code> ; (2) an edgelist in the form of a two-column dataframe; (3) an <code>igraph</code> object. |
| s         | numeric: Proportion of edges to retain, $0 < s < 1$ ; smaller values yield sparser graphs                                                                                                                     |
| class     | string: the class of the returned backbone graph, one of <code>c("original", "matrix", "Matrix", "igraph", "edgelist")</code> . If "original", the backbone graph returned is of the same class as U.         |
| narrative | boolean: TRUE if suggested text & citations should be displayed.                                                                                                                                              |

## Value

An unweighted, undirected, unipartite graph of class `class`.

## References

- package: Neal, Z. P. (2022). backbone: An R Package to Extract Network Backbones. *PLOS ONE*, 17, e0269137. doi:[10.1371/journal.pone.0269137](https://doi.org/10.1371/journal.pone.0269137)
- model: Karger, D. R. (1999). Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24, 383-413. doi:[10.1287/moor.24.2.383](https://doi.org/10.1287/moor.24.2.383)

## Examples

```
U <- igraph::erdos.renyi.game(60, .5)
plot(U) #A dense graph
sparse <- sparsify.with.skeleton(U, s = 0.25, narrative = TRUE)
plot(sparse) #A sparser graph
```

# Index

backbone, 2  
backbone-package (backbone), 2  
backbone.extract, 3  
backbone.extract(), 7, 10, 12, 14, 16, 19, 22, 24, 27  
backbone.suggest, 4  
backbone.suggest(), 3  
bicm, 5, 20, 27  
bicm(), 3  
  
disparity, 6  
disparity(), 2, 29  
  
fastball, 8  
fastball(), 3, 10  
fdsm, 9  
fdsm(), 2  
fixedcol, 11  
fixedcol(), 2  
fixedfill, 13  
fixedfill(), 2  
fixedrow, 15  
fixedrow(), 2  
  
global, 17  
global(), 2  
  
igraph, 5, 7, 9, 12, 13, 15, 17, 18, 21, 23, 26, 28, 30–39  
  
lans, 18  
lans(), 2  
logit, 20, 27  
loglikelihood\_prime\_bicm, 5  
  
Matrix, 5, 7, 9, 12, 13, 15, 17, 18, 21, 23, 26, 28, 30–39  
mlf, 21  
mlf(), 2  
  
osdsm, 23  
  
osdsm(), 2  
  
p.adjust, 3, 7, 10, 12, 14, 15, 19, 21, 23, 26  
pb, 25  
  
sdsm, 20, 26  
sdsm(), 2, 6  
sparsify, 28  
sparsify(), 3, 30–39  
sparsify.with.geometric, 30  
sparsify.with.geometric(), 3, 29  
sparsify.with.gspar, 31  
sparsify.with.gspar(), 3, 29  
sparsify.with.hypergeometric, 32  
sparsify.with.hypergeometric(), 3, 29  
sparsify.with.jaccard, 33  
sparsify.with.jaccard(), 3, 29  
sparsify.with.localdegree, 34  
sparsify.with.localdegree(), 3, 29  
sparsify.with.lspar, 35  
sparsify.with.lspar(), 3, 29  
sparsify.with.meetmin, 36  
sparsify.with.meetmin(), 3, 29  
sparsify.with.quadrilateral, 37  
sparsify.with.quadrilateral(), 3, 29  
sparsify.with.simmelian, 38  
sparsify.with.simmelian(), 3, 29  
sparsify.with.skeleton, 39  
sparsify.with.skeleton(), 3, 29  
  
tomatrix, 4  
trials.needed(), 10, 24